Pose and Depth Estimation: 3D Reconstruction using Videos

Manash Pratim Barman
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
`mbarman@andrew.cmu.edu`

Tejas Mehta
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
`tnmehta@andrew.cmu.edu`

Subhadra Gopalakrishnan
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
`subhadrg@andrew.cmu.edu`

Abhishek Mahajani
Heinz School of Information
Carnegie Mellon University
Pittsburgh, PA 15213
`aum@andrew.cmu.edu`

Abstract

As a way of emulating the human visual system, there have been several attempts and approaches to visualize our 3D world from its 2D projections (images and videos). While multiple classical and geometric methods exist, they are computationally expensive and time consuming. Deep learning based pose and depth estimation methods have recently shown highly promising results in this field. We propose a novel architecture for depth estimation – ResDepth, that can be trained using stereo pairs in a self-supervised setting. Our goal was to create an architecture that is not very complex (both in terms of design and number of parameters), yet robust enough. Our model gives satisfactory results and tends to perform better than the baseline model.

1 Introduction

We, as humans, can discern a single image and perceive a rough depth estimate of the scene. This is due to the fact that we developed a rich knowledge of objects and the structural understanding of the world through the accumulation of years of data. We are also good at estimating our position relative to the world (ego-motion) in a short time.

Depth estimation has fascinated researchers due to its myriad applications which include autonomous driving, augmented reality, virtual tourism, 3D reconstruction of a scene, tasks like synthetic de-focus and depth-based visual effects to name a few.

Traditional methods of visual odometry involved matching of points, 44 features or patches from one frame to another and the estimation of the ego-motion of the camera from it. These methods depend heavily on feature extraction and matching algorithms. On the other hand, depth estimation from a single image using classical methods is a non-trivial task.

With the advent and subsequently, the performance of deep learning architectures like convolutional neural networks in complicated vision related tasks led researches to adopt these approaches to depth and pose estimation as well. The learning-based methods can help with solving the depth estimation and mapping problem. Because the mapping problem is heavily interlinked with the ego-motion problem, many researchers have tried to estimate pose and depth simultaneously using deep learning.

However, deep learning algorithms does have certain limitations. Many learning-based methods deal with complications like the need for intrinsic parameters of the camera, which may not always be available, or failing to work on dynamic scenes where both the camera and scene are in motion. Another hurdle is the lack of varied, labeled data-sets. The existing ones are quite limited to a particular scene or environment, and hence generalizing over an unknown environment is a challenge. This led researchers to explore unsupervised / self-supervised depth estimation where labeled data is not necessary.

In this work, we re-implemented a state of the art architecture for depth estimation proposed by [1], published in ICCV 2019. Further, we propose a novel architecture for depth estimation using stereo vision in a self-supervised setting which performs better than the baseline [1] model. Our baseline model involved re-implementing the stereo and monocular models for depth and pose estimation proposed by [1]. Their approach involved using an input image at multiple scales and computing disparity (and depth) of an image using multiple versions of that image. For instance, their stereo model uses a total of 8 images to compute disparity and loss. These 8 images include 4 multi-scale versions of the target image (input image or the left stereo image) as well as 4 such versions for the right stereo pair of the input image which are used to compute reconstruction error. All the results and implementation details of this baseline model is discussed in the Appendix - Section 8.

Implementing the baseline model helped us understand that their proposed architecture is over-parameterized and complicated. We also found that their approach lacks in the depth estimation of homogeneous objects. We explored the possibility of developing simpler architectures that achieves similar results without complicated procedures like multiscaling. In our work, we propose a novel architecture (ResDepth) for depth estimation using stereo vision in a self-supervised setting. We put forward the idea of using a Gaussian filter for computing SSIM loss as mentioned in [13]. Such an implementation simplifies the model architecture with less parameters and ameliorates the problem of homogeneity to an extent.

2 Related Work

Traditional methods of depth estimation from stereo images [2] usually estimate matching correspondences from stereo pairs and calculate the disparity between them which is then used to calculate the depth. These dense stereo correspondence algorithms utilized geometry to estimate constraints which then could be used to solve the problem.

Estimating depth from single images, while being geometrically infeasible, has been attempted by multiple supervised and unsupervised deep learning methods. [3] is one of the very first supervised learning approaches to this problem. They estimate a conditional distribution of depth from the monocular image features using a hierarchical, multi-scale, Markov Random field.

Currently, researchers in this domain are exploring self-supervised / unsupervised methods, primarily because of the dearth of a wide variety of labelled data. Some methods use consecutive time frames to estimate the pose which they then use as supervisory signal to train the model [4]. Another way to approach this problem is to use stereo pairs to train the model [5]. For this method, along with the stereo pairs during training, the baseline should also be known. This stereo supervision method has been shown to perform better than monocular training usually. This is due to inconsistencies when dealing with consecutive video frames like non rigid scene motion. Usually, a sort of motion segmentation mask is used to mask out moving objects [4].

[6] tries to close the gap between stereo and monocular supervision by estimating the pose CNN using geometry based Direct Visual Odometry methods arguing that the additional model for estimating the pose is the cause of the weakness of monocular supervision.

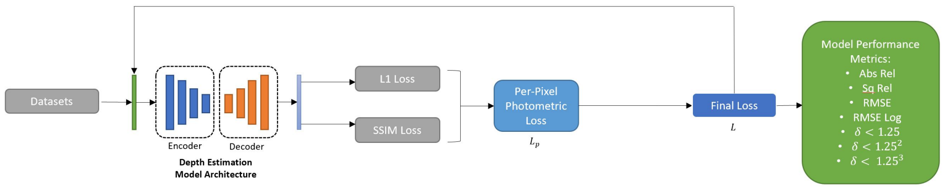
[7] works with both monocular and stereo supervised training and introduces some important additions like an automasking approach to ignore the pixels that violate the static scene constraint and a minimum re-projection loss to account for occluded correspondences. [8] replaces the image classification based ResNet architectures for encoders and decoders with models more suited for fine grained discrimination for depth prediction. They also try to use the velocity of the camera when available to solve the scale ambiguity in monocular training.

Many methods of unsupervised training using monocular video have been attempted. [9] uses an estimated depth map for pose estimation. Recurrent units are interleaved in the encoder which are then fed to a decoder network to generate a depth map. This depth map is concatenated with the RGB image and fed into a CNN to estimate the pose. [10] is an unsupervised approach using Generative Adversarial Networks for both depth and pose estimation. The generator network maps the input image to a depth map, and the discriminator network receives a view reconstruction using the depth and pose estimation which requires the camera intrinsics. It then outputs a probability of whether the reconstruction is similar to the original image.

3 Approach

Our model, ResDepth, is an overhaul of the baseline model proposed by [1]. As mentioned previously, the details of the baseline implementation can be found in the Appendix - Section 8. We have used KITTI Dataset for implementing the stereo-based self supervised training. Images from the dataset were passed into model architecture with encoder and decoder to output a disparity map. The final loss is computed using a combination of L1 and SSIM losses between the target and the output. This loss is then back-propagated to update the parameters of the model. The complete approach of our ResDepth model is explained in Figure 1.

Figure 1: Overview of ResDepth Model for Stereo Training



3.1 Datasets

Our implementation of the ResDepth model involves the use of the KITTI Raw Dataset. It contains a suite of vision tasks built using an autonomous driving platform and the full benchmark contains tasks like stereo, optical flow, visual odometry, etc.

Discussing the data format, each sensor stream is compiled in a single folder with timestamp files up to nanosecond precision. Four synchronized cameras at about 10 Hz with respect to the velodyne laser scanner were used to collect the image and velodyne data. The image data (1382 x 512 pixels) contains left rectified grayscale image sequence, right rectified grayscale image sequence, left rectified color image sequence, and right rectified color image sequence. While, the velodyne data in the 'velodyne points' contains 4 values - the first three corresponding to x, y and z coordinates and the last value corresponding to the information about reflectance. Along with the above data, we also have access to sensor calibration data to perform model training.

3.1.1 Dataloader Implementation

We follow the data split of Eigen et al. [11] using the preprocessing techniques described by Zhou et al. [12] to remove static frames. It consists of 39810 training items and 4424 validation items. Pre-processing of the images include adjusting brightness, contrast, saturation and hue of the input images. The implementation employs a data-loader that returns a dictionary for a single training item consisting keys for raw color images, augmented images, the corresponding stereo image, camera intrinsic matrix its inverse, camera extrinsic and ground truth depth maps on the single original scale as opposed to the technique of multi-scaling described in [1]. In our implementation, we have continued to use this representation as it is an efficient way to extract the multiple values required for training. The ground truth depth maps are generated from the velodyne points and sensor calibration files.

For our proposed model, we only use the augmented color images of a resolution of 640 X 192 and its corresponding stereo pairs and the camera intrinsics.

Table 1: Encoder Architecture - ResDepth

ResNet 18 Encoder		
Layer Name	Architecture	Activation
conv1	7x7,64,stride=2	ReLU
layer1	$\begin{bmatrix} 3\times3,64 \\ 3\times3,64 \end{bmatrix}\times 2$	ReLU
layer2	$\begin{bmatrix} 3\times3,128 \\ 3\times3,128 \end{bmatrix}\times 2$	ReLU
layer3	$\begin{bmatrix} 3\times3,256 \\ 3\times3,256 \end{bmatrix}\times 2$	ReLU
layer4	$\begin{bmatrix} 3\times3,512 \\ 3\times3,512 \end{bmatrix}\times 2$	ReLU

3.2 Model Architecture

Our ResDepth model consists of an encoder and a decoder. For our encoder, we use a pretrained ResNet 18 model. Pre-trained models converges much faster and performs better, especially in vision related tasks. Hence, the use of pre-trained models (transfer learning) is prevalent in this area of research.

The encoder (Table 1) takes in one of the images of the stereo pair and we extract the outputs of the model at five different layers namely after the first convolution layer (with batch normalization and ReLU activation) and the output at each ResNet block. We use these features in our decoder to compute disparity (and depth). Each ResNet block reduces the size of the image by a factor of 2.

Our decoder (Table 2) consists of 4 layers and the output layer. In each of the first 4 layers, we use the outputs of the encoder in reverse order, two at a time by concatenating them. For example, in the first decoder layer, we use a concatenation of the output of layer 4 and layer 3 of the encoder as its input. Since the outputs of two different layers of the encoder are of different sizes, we upsample the smaller output by a factor of 2 to match the larger output. The upsampling is done using the interpolate function of Pytorch in the bilinear mode as it is locally sub-differentiable. Each layer in the decoder consists of two convolution layers with a kernel size of 3, stride and padding of 1. The convolution layers are activated using LeakyRelu with a negative slope of 0.2. The choice of LeakyReLu is based on empirical evidence of results. Each of the layers contain different number of filters. The final layer is a convolution layer with 1 filter and a kernel size of 3. This layer is activated with sigmoid activation to get the disparity values between 0 and 1.

The major difference between the baseline model in [1] and our model is that we do not use multiple scales of an image. We use the image at the original scale (640×192). In comparison, [1] uses 8 images (4 for left stereo pair and 4 for right stereo pair) to compute disparity (depth) of a single image. This reduces the complexity of our model significantly.

3.3 Loss Functions

To compute loss, depth values predicted by the model and the corresponding right stereo pair of the input image along with its camera intrinsics are used to reconstruct the input image. Loss is measured by the quality of the reconstructed image compared to the input image. The reconstruction of images require non-trivial knowledge of core computer vision. Hence, we use the implementation of [1] for that part.

Commonly used loss functions for depth estimation include L1 loss, Smooth loss and SSIM loss. The L1 loss measures the total of all the absolute errors between each pixels of two images. The Smooth loss is used to prevent the estimated depths from shrinking. SSIM loss helps in measuring the similarity between the two images. It is computed based on the three comparison measurements between the images x and y: luminance (l), contrast (c) and structure (s). The individual comparison functions are:

Table 2: Decoder Architecture - ResDepth

Decoder		
Layer Name	Architecture	Activation
layer1	Upsample $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 1$	LeakyReLU
layer2	Upsample $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 1$	LeakyReLU
layer3	Upsample $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 1$	LeakyReLU
layer4	Upsample $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 1$	LeakyReLU
Output layer	3x3, 1, stride=1, padding=1	Sigmoid

$$\mathrm{l}(\mathbf{x},\mathbf{y}) = \frac{2\mu_{\mathbf{x}}\mu_{\mathbf{y}}+c_1}{{\mu_{\mathbf{x}}}^2+{\mu_{\mathbf{y}}}^2+c_1}$$

$$\mathrm{c}(\mathbf{x},\mathbf{y}) = \frac{2\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}+c_2}{{\sigma_{\mathbf{x}}}^2+{\sigma_{\mathbf{y}}}^2+c_2}$$

$$\mathrm{s}(\mathbf{x},\mathbf{y}) = \frac{\sigma_{\mathbf{xy}}+c_3}{(\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}+c_3}$$

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha} \cdot [c(\mathbf{x},\mathbf{y})]^{\beta} \cdot [s(\mathbf{x},\mathbf{y})]^{\gamma}$$

where $l(\mathbf{x},\mathbf{y}), c(\mathbf{x},\mathbf{y}), s(\mathbf{x},\mathbf{y})$ are the luminance, contrast and structural comparisons between the 2 images x, y and $\alpha > 0, \beta > 0, \gamma > 0$ are parameters used to adjust the relative importance.

Our implementation of the loss computation significantly differs from the baseline model [1], detailed explanation of which can be found in Appendix - Section 8. They use a combination of all the losses - L1, Smooth and SSIM. For our proposed model, we compute the per-pixel photo-metric loss using only a combination of L1 and SSIM loss. In addition, we calculate the SSIM loss for images using a sliding Gaussian window of size 11×11. The window can be displaced pixel-by-pixel on the image to create an SSIM quality map of the image. This showed improvements in the results compared to the multi-scale SSIM (using Average Pooling with a kernel size of 3) used by [1] which is conducted over multiple scales through a process of multiple stages of sub-sampling, reminiscent of multi-scale processing in the early vision system. The several changes in the computation of the loss proposed by us significantly improve the predicted depth quality.

3.4 Model Performance

The performance of models for depth estimation are measured by two categories of metrics namely error and accuracy under a threshold. There are 4 different error metrics and 3 different accuracy metrics. A model is expected to give low error values and high accuracy values. In our work, we evaluate our model's performance using the same conventional metrics followed in this domain which are described below.

1) **Errors Metrics**:

1. Absolute Relative Error: It is the magnitude of the difference between the ground truth depth and the predicted depth divided by the magnitude of the ground truth depth.
2. Squared Relative Error: It is the squared value of Absolute Relative Error.
3. Root Mean Squared Error: It is the squared root of the variances of the error between the ground truth and predicted depth values. It tells us how close the ground truth values are from the predicted values.
4. Root Mean Squared Error Log: It is the Logarithmic value of RMSE

2) **Accuracy under a Threshold Metrics**: It denotes the maximum relative error of a model under a predefined threshold. These values are loosely interpreted as accuracy of a model and is expected to be high.

1. Threshold $\delta < 1.25$
2. Threshold $\delta < 1.25^2$
3. Threshold $\delta < 1.25^3$

4 Results

The model is evaluated using the above conventional metrics. Below is a table comparing our results with other stereo implementations.

4.1 Quantitative Results

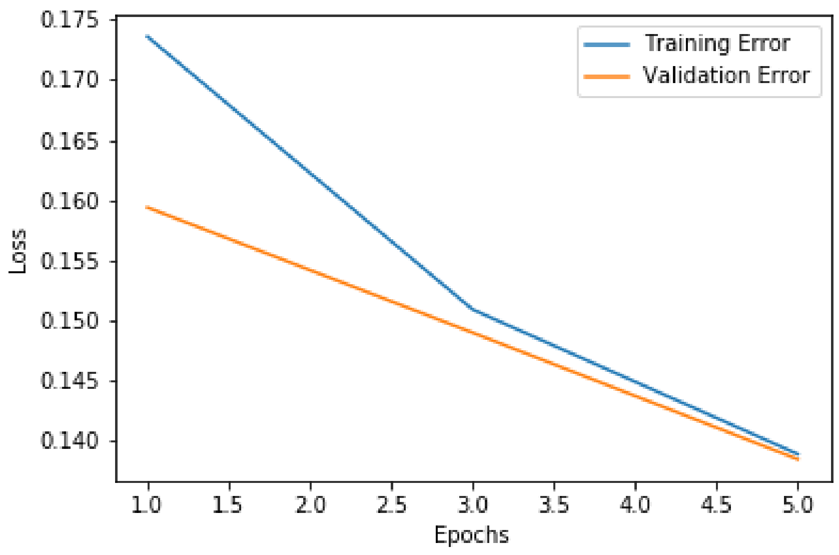
Table 3: Comparison of our method to existing methods on KITTI 2015 using the Eigen split.

Method	Train	Abs Rel	Sq Rel	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Garg [12]†	S	0.152	1.226	5.849	0.246	0.784	0.921	0.967
Monodepth R50 [15]†	S	0.133	1.142	5.533	0.230	0.830	0.936	0.970
StrAT [43]	S	0.128	1.019	5.403	0.227	0.827	0.935	0.971
3Net (R50) [50]	S	0.129	0.996	5.281	0.223	0.831	0.939	0.974
3Net (VGG) [50]	S	0.119	1.201	5.888	0.208	0.844	0.941	0.978
SuperDepth + pp [47]	S	0.112	0.875	4.958	0.207	0.852	0.947	0.977
Monodepth2	S	0.109	0.873	4.960	0.209	0.864	0.948	0.975
Monodepth2	MS	0.106	**0.806**	**4.630**	**0.193**	0.867	0.958	0.980
Resdepth	S	**0.101**	0.86	4.68	**0.193**	**0.9007**	**0.96**	**0.9802**

Note: The best results in each category are in bold, while the second best are underlined. All results here are presented without post-processing. The results presented in the table for the Monodepth2 model are the original results presented by the authors of [1] and not from our re-implementation. Our re-implementation results can be found in Appendix - Section 8.

From the above table, we can clearly see that our model outperforms the baseline Monodepth2 model for the Stereo training method. It is surprising to see that our model performed better than the Monodepth2 model using both Mono and Stereo training method on most of the metrics. We also present the trend of the training and validation error plot up to 5 epochs in Figure 2.

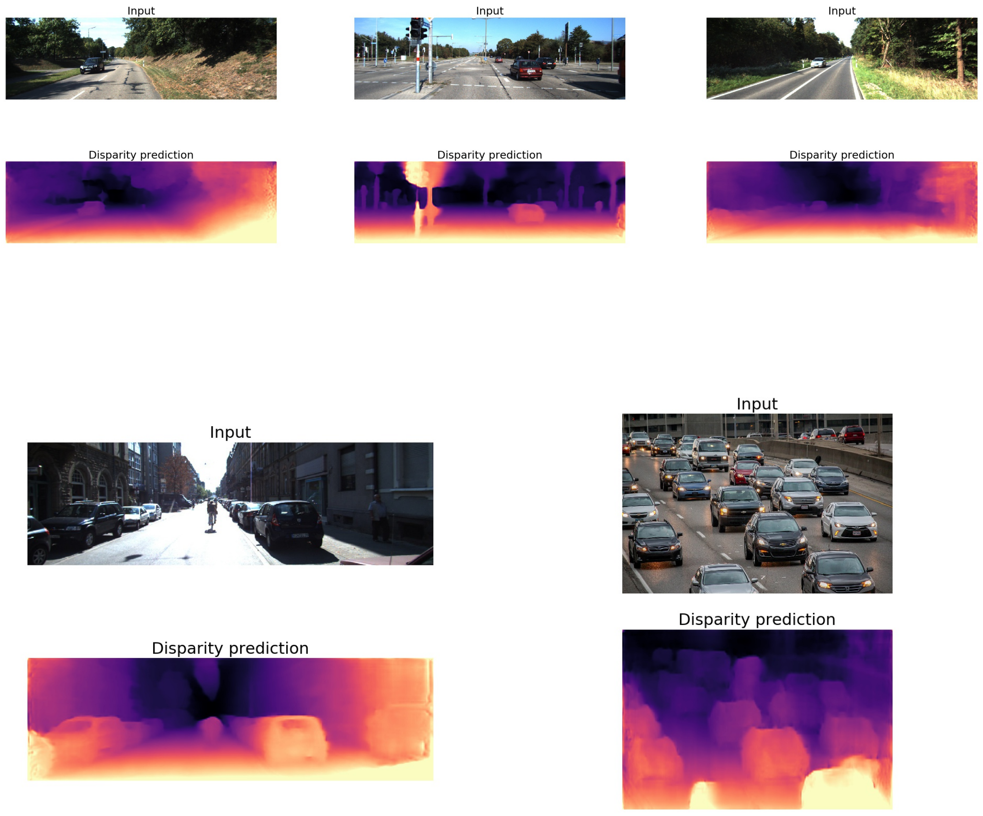
Figure 2: Training and Validation Error Plot for ResDepth



4.2 Qualitative Results

The following section show the resulting disparity maps obtained from our model - ResDepth on single input images in the Table 3. We also show the comparison between the outputs from the baseline architecture and our architecture in the Table 4. We can clearly see over here that our model is able to predict better disparity maps in the areas of high homogeneity. One of the other area of improvement we focused on was to reduce depth errors caused by reflections from glass surfaces.

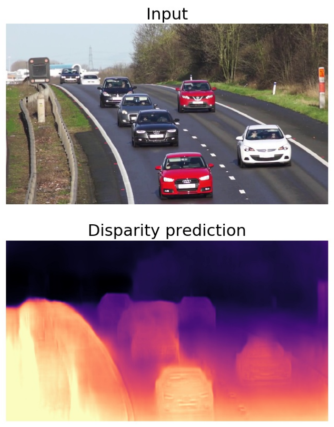
Table 3: Disparity Maps on Single Images using ResDepth Model



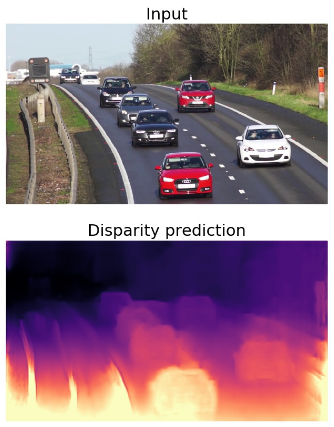
To test the robustness of our architecture on the real-world scenarios, we tested it on a YouTube video of car driving in Toronto and created a video giving the disparity maps. The link to the video of Disparity Maps generated is given in the section 6.

From the above observations and results we can infer that ResDepth model is also able to perform better without the smooth loss and multi-scaling as suggested by the paper. Instead of increasing the model complexity to improve results, we observe that significant improvements could be achieved by changing the loss functions and underlying architecture. Further, using concatenation as skip connection in the decoder helped us in reducing its depth and might have led to better feature reuse as well as better propagation of gradients compared to using summation as skip connection which might pollute feature maps especially for a shallower network. Further investigation is required to validate this conjecture.

Table 4: Comparison of Disparity Maps - Baseline v/s ResDepth



Baseline Architecture



Our ResDepth Architecture

5 Conclusion

We propose a novel architecture - ResDepth to estimate depth using stereo vision and self-supervised deep neural networks. Our model leverages the power of transfer learning similar to the other state-of-the-art models. We improve on the performance of [1] for stereo vision and show that simpler architectures can perform as well as complicated over-parameterized architectures if not better. Besides, our work also involved re-implementation of the state of the art architecture [1] for depth estimation published in ICCV 2019 using both stereo and monocular vision in a self-supervised setting upon which we have improvised. We used some parts of their code that involved knowledge of core computer vision to re-implement the models. Future work involves extension of ResDepth for monocular vision, experimenting with other loss functions that might improve the performance and exploring on how a comparatively shallower network can achieve comparable performance to a deeper network.

6 Links

GitHub: `https://github.com/OkBoomer11785/Project`
Disparity Map Video: `https://www.youtube.com/watch?v=XAEVeaF7Utk`
Project Video: `https://youtu.be/YCgQkv8tWfE`

7 References

[1] Godard, Clement, Ois in Mac Aodha, Michael Firman, and Gabriel Brostow. “Digging Into Self-Supervised Monocular Depth Estimation.” 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019. https://doi.org/10.1109/iccv.2019.00393.

[2] Scharstein, D., R. Szeliski, and R. Zabih. “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms.” Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001), n.d. https://doi.org/10.1109/smbv.2001.988771.

[3] Saxena, Ashutosh, Sung H. Chung, and Andrew Y. Ng. “3-D Depth Reconstruction from a Single Still Image.” International Journal of Computer Vision 76, no. 1 (2007): 53–69. https://doi.org/10.1007/s11263-007-0071-y.

[4] Zhou, Tinghui, Matthew Brown, Noah Snavely, and David G. Lowe. “Unsupervised Learning of Depth and Ego-Motion from Video.” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. https://doi.org/10.1109/cvpr.2017.700.

[5] Godard, Clement, Oisin Mac Aodha, and Gabriel J. Brostow. “Unsupervised Monocular Depth Estimation with Left-Right Consistency.” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. https://doi.org/10.1109/cvpr.2017.699.

[6] Wang, Chaoyang, Jose Miguel Buenaposada, Rui Zhu, and Simon Lucey. “Learning Depth from Monocular Videos Using Direct Methods.” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018. https://doi.org/10.1109/cvpr.2018.00216.

[7] Godard, Clement, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. “Digging Into Self-Supervised Monocular Depth Estimation.” 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019. https://doi.org/10.1109/iccv.2019.00393.

[8] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Adrien Gaidon. “PackNet-SfM: 3D Packing for Self-Supervised Monocular”. Accessed April 8, 2020. https://arxiv.org/pdf/1905.02693v2.pdf.

[9] Wang, Rui, Stephen M. Pizer, and Jan-Michael Frahm. "Recurrent neural network for (un-)supervised learning of monocular video visual odometry and depth." In Proceedings of the IEEEConference on Computer Vision and Pattern Recognition, pp. 5555-5564. 2019

[10] Almalioglu, Yasin, Muhamad Risqi U. Saputra, Pedro PB de Gusmao, Andrew Markham, and Niki Trigoni. "GANVO: Unsupervised deep monocular visual odometry and depth estimation withgenerative adversarial networks." In 2019 International Conference on Robotics and Automation(ICRA), pp. 5474-5480. IEEE, 2019 cl[11] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In ICCV, 2015

[12] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised learning of depth and ego-motion from video. In CVPR, 2017

[13] Alhashim, I. and Wonka, P., 2018. High quality monocular depth estimation via transfer learning. arXiv preprint arXiv:1812.11941.

8 Appendix - Baseline Implementation and its Approach

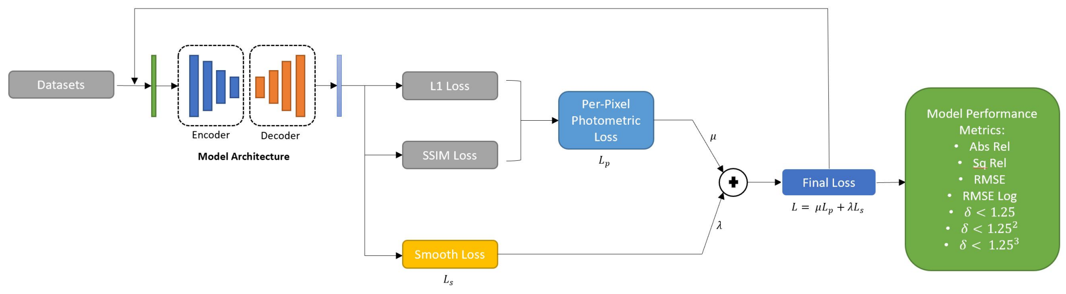
Baseline re-implemented is the research paper - "Digging Into Self-Supervised Monocular Depth Estimation: (ICCV 2019) - [1]. This re-implementation works for both stereo and monocular training with overall approach discussed in the Figure 3 and Figure 4 respectively.

8.1 Baseline Model Architecture

The model used for depth estimation in the baseline is an encoder decoder pair. The encoder (as described in the Table 5) is based on a ResNet-18 architecture with a convolution layer followed by 4 residual blocks. Using a deep convolutional model for encoding provides an improvement in capturing localized information along with additional abstract features such as geometry of the shapes. The output of this encoder, along with the outputs at the intermediate layers are fed into the decoder layers.

The decoder network shown in Table 6 has five upsampling convolution blocks where the output of each block is added to the residual connections from the encoder, basically following a U-Net like architecture. Each layer has 2 Conv Blocks with a padding of 1 and convolved with a kernel size of 3 and stride 1. This is followed by a Conv3x3 Block which is similar to Conv Block except that it uses reflection padding instead of zero padding. The scaling has been done at the end of each layer to identify large objects by using a deeper network to identify larger features similar to results after gaussian blur with a higher standard deviation. The decoder converts the encoder output into a feature map. The depth decoder has Sigmoids at the output at each level with exponential linear units (ELU) instead of the commonly used rectified liner units (ReLU). We are computing the depth map pixel wise using a formula $D = \frac{1}{(a\sigma+b)}$ where σ is the sigmoid of the disparity value at the output layer. Here a and b are used to limit depth values between 0.1 and 100 units. The detailed architecture for the encoder and decoder is shown below:

Figure 3: Overview of Baseline Model for Stereo Training



8.2 Pose estimation - Monocular Training

The 3D depth estimation model trained using monocular videos requires a separate deep learning network, as shown in Figure 4, for estimating the pose between 2 frames. Two consecutive frames are sent to the pose encoder that predicts the relative translation and rotation of the camera between them. This is then used to backproject the predicted depth into a predicted frame that is used to calculate the reprojection loss. The pose model consists of an encoder and decoder. The encoder is also based on a ResNet - 18 architecture and the decoder is made up of four convolutional layers. The pose network predicts the rotation and translation between the frames.

8.3 Loss Functions

Multiple approaches have used a multiscale re-projection loss for training [5], where each decoder output is used to compute the loss. However, Monodepth2 argues that upsampling the intermediate outputs to the input image resolution improves the model learning to reconstruct the high resolution

Table 5: Encoder Architecture - For both Pose and Depth

ResNet Encoder for depth		
Layer Name	Architecture	Activation
conv1	7x7,64,stride=2	ReLU
3x3 MaxPool stride=2		
layer1	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	ReLU
layer2	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	ReLU
layer3	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	ReLU
layer4	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	ReLU

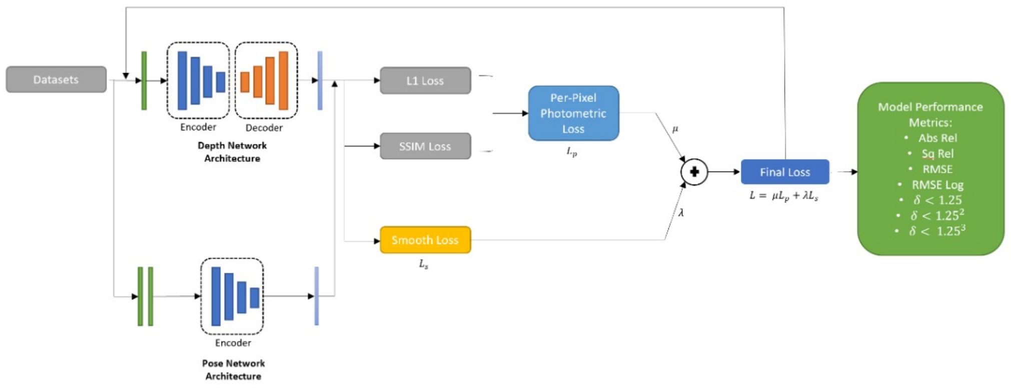
Table 6: Decoder Architecture - Depth

Decoder architecture		
Layer Name	Architecture	Activation
upconv$_{40}$	3x3,256,stride=1	ELU
Scale up x2		
upconv$_{41}$	5x5,256,stride=1	ELU
upconv$_{30}$	3x3,128,stride=1	ELU
Scale up x2		
upconv$_{31}$	5x5,128,stride=1	ELU
dispconv$_3$	5x5,1,stride=1	ELU
upconv$_{20}$	3x3,64,stride=1	ELU
Scale up x2		
upconv$_{21}$	5x5,64,stride=1	ELU
dispconv$_2$	5x5,1,stride=1	ELU
upconv$_{10}$	3x3,32,stride=1	ELU
Scale up x2		
upconv$_{11}$	5x5,32,stride=1	ELU
dispconv$_1$	5x5,1,stride=1	ELU
upconv$_{00}$	3x3,16,stride=1	ELU
Scale up x2		
upconv$_{01}$	5x5,16,stride=1	ELU
dispconv$_0$	5x5,1,stride=1	ELU

Table 7: Decoder Architecture - Pose

Decoder architecture		
Layer Name	Architecture	Activation
conv$_1$	1x1,256,stride=1	ReLU
conv$_2$	3x3,256,stride=1	ReLU
conv$_3$	3x3,256,stride=1	ReLU
conv$_4$	1x1,6,stride=1	ReLU

Figure 4: Overview of Baseline Model for Monocular Training



output as accurately as possible. The total loss is a weighted sum of the multiscale reprojection and smoothness loss.

Furthermore, we are computing the photometric error function pe using L1 and SSIM. SSIM helps in providing an improved image similarity assessment over MSE. It takes into account the luminance, contrast, and structure of images to compute Normalized Cross-Correlation (NCC).

$$\text{SSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha} \cdot [c(\mathbf{x},\mathbf{y})]^{\beta} \cdot [s(\mathbf{x},\mathbf{y})]^{\gamma}$$

where $l(\mathbf{x},\mathbf{y}), c(\mathbf{x},\mathbf{y}), s(\mathbf{x},\mathbf{y})$ are the luminance, contrast and structural comparisons between the 2 images x, y and $\alpha > 0, \beta > 0, \gamma > 0$ are parameters used to adjust the relative importance.

The smoothness loss penalizes the model for high gradients in the disparity image when the image does not have high gradients (edges) at the same point.

$$L_s = |\partial_x d_t^*| \, e^{-|\partial_x I_t|} + |\partial_y d_t^*| \, e^{-|\partial_y I_t|}$$

where $d_t^* = \frac{d_t}{d_t}$ is mean normalized inverse depth and I_t is the image at current timestep t.

Finally, the function loss is computed using a weighted average of smoothness loss and re-projection loss using the formula:-

$$L = \mu L_p + \lambda L_s \text{ where } \mu = 1 \text{ and } \lambda = 0.001$$

Monocular Training also requires a few additional elements like minimum reprojection loss and automasking to account for occlusion and non rigid scene motion.

Choosing the minimum reprojection loss helps in minimizing losses because of occlusions and disocclusions. This helps in matching each pixel to view only in which it is visible. This gives better results as compared to mean reprojection loss. Instead of matching the occluded pixels to the current view, the loss function accounts for those pixels in the previous or next view when they are visible.

$$L_p = \min_{t'} pe\left(I_t, I_{t' \to t}\right)$$

where $pe\left(I_t, I_{t' \to t}\right)$ is the photometric error between the image at timestep t and the adjacent frames at time $t+1$ and $t-1$.

This helps in reducing the artifacts at image boundaries and sharpens the occlusions boundaries resulting in better accuracy.

One particular challenge while training with monocular videos is moving objects. If an object moves at the same speed as a camera, the estimated disparity becomes zero and causes areas of infinite depth while predicting. This is avoided by using a simple automasking approach that masks out pixels with no relative motion in consecutive frames while calculating the reprojection loss.

8.4 Results - Baseline Implementation

8.4.1 Results - Stereo Baseline

This section deals with the results of the stereo training on the baseline model implemented (as discussed in the Figure 3). The trend of the training and validation losses for 10 epochs can be found in Figure 5. Besides, the metrics performance are shown in the Table 8. Additionally, we also verified and predicted disparity maps on single images after Epoch 1 and Epoch 10 of training as shown in the Table 10.

8.4.2 Results - Monocular Baseline

This section deals with the results of the monocular training on the baseline model implemented (as discussed in the Figure 4). The trend of the training and validation losses for 10 epochs can be found in Figure 6. Besides, the metrics performance are shown in the Table 9.

8.4.3 Ablation Study

We performed ablation study in the baseline implementation of [1] for the different values of α used to combine the L1 and SSIM loss as per the equation and the results are in Table 11:

$$\mathrm{pe}(I_a, I_b) = \frac{\alpha}{2}(1 - SSIM(I_a, I_b)) + (1 - \alpha)||I_a, I_b||_1$$

Table 8: Model Performance Metrics - Stereo Training

Epoch	Abs Rel	Sq Rel	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
1	0.1235	1.2281	5.3235	0.2083	0.8673	0.9538	0.9796
2	0.1081	1.0329	5.0307	0.2026	0.8869	0.9556	0.9790
3	0.1040	1.0307	4.9197	0.1962	0.8964	0.9593	0.9806
4	0.1030	1.1374	5.1299	0.2007	0.8993	0.9589	0.9792
5	0.1073	1.1496	5.2063	0.2018	0.8979	0.9587	0.9789
6	0.0949	0.9789	4.7864	0.1947	0.9102	0.9614	0.9797
7	0.0986	0.9855	4.7735	0.1952	0.9107	0.9609	0.9796
8	0.0945	1.0353	4.9268	0.1951	0.9092	0.9606	0.9794
9	0.0957	1.1313	5.0547	0.1950	0.9109	0.9615	0.9796
10	0.0913	0.9676	4.7219	0.1923	0.9169	0.9624	0.9799

Table 9: Model Performance Metrics - Monocular Training

Epoch	Abs Rel	Sq Rel	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
1	0.1593	1.0713	6.0772	0.2506	0.7498	0.9179	0.9722
2	0.1629	1.2199	5.5326	0.2338	0.7781	0.9395	0.9789
3	0.1504	1.0658	5.0681	0.2224	0.8113	0.9446	0.9796
4	0.1429	0.9468	4.8853	0.2244	0.8115	0.9471	0.9802
5	0.1391	0.9024	4.8480	0.2174	0.8269	0.9462	0.9817
6	0.1395	0.9742	4.8989	0.2206	0.8317	0.9445	0.9804

Figure 5: Trend of Training and Validation Loss for 10 Epochs - Stereo Training

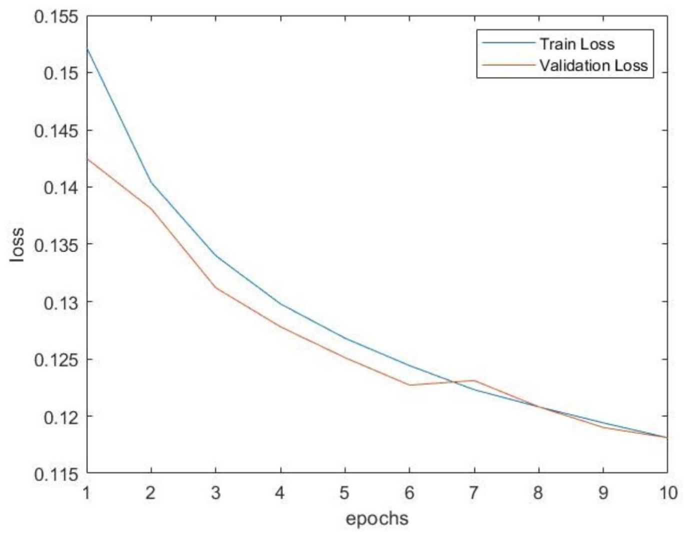


Table 10: Disparity Map Prediction for Single Images - Stereo Training

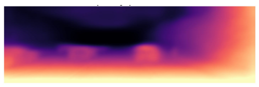
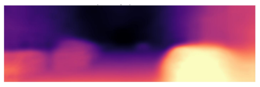
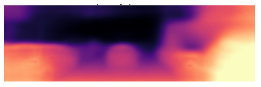
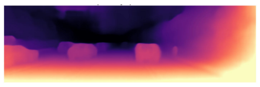
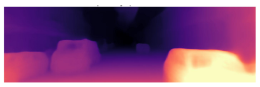
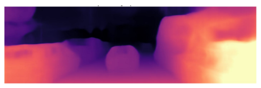
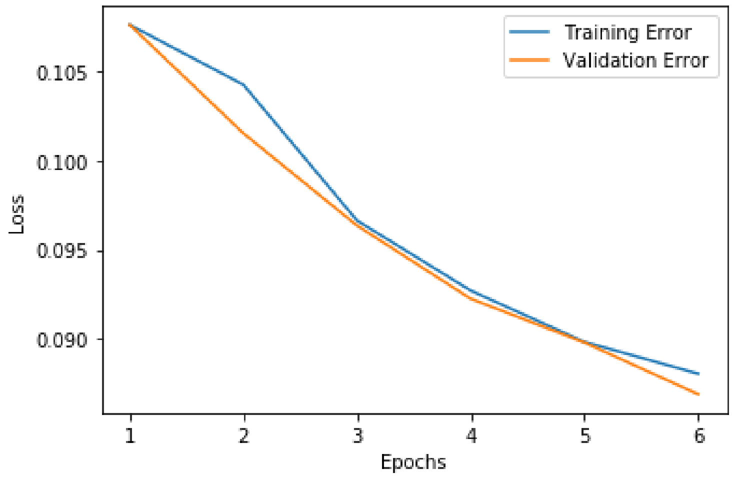
Input Image			
Disparity Map after Epoch 1			
Disparity Map after Epoch 10			

Table 11: Ablation study with varying α values

α	Abs Rel	Sq Rel	RMSE	RMSE Log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
0.50	0.12417	1.38552	5.61875	0.21063	0.87248	0.95333	0.97796
0.70	0.11604	1.13415	5.10498	0.20653	0.88375	0.95378	0.97749
0.85	0.12083	1.21563	5.26317	0.20875	0.87856	0.95329	0.97778
0.90	0.11298	1.03042	4.92409	0.20196	0.88507	0.95593	0.97926
0.95	0.10675	1.01742	4.90986	0.20134	0.88757	0.95615	0.97889

Figure 6: Trend of Training and Validation Loss for 6 Epochs - Monocular Training


Training Error
Validation Error
Loss
Epochs